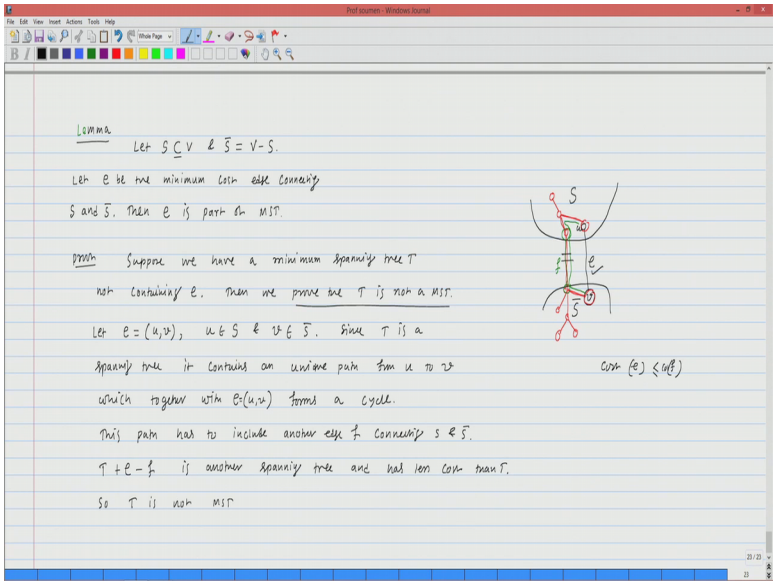
Graph Theory
Prof. Soumen Maity
Department of Mathematics
Indian Institute of Science Education and Research, Pune

Lecture - 06
Part 2
Minimum Spanning Trees (cont.)

Welcome to the second part of 6th lecture. So, in the first part we talked about a one lemma that suppose s is a subset of the vertex set v and s complement is v minus s. And let e be the minimum cost edge connecting s, s complement means this is a minimum cost edge of the cut s s complement then e must be part of minimum Spanning Tree.

(Refer Slide Time: 00:54)



So, this is a important result and this sort of signify why Prim's algorithm is correct. Suppose we have a minimum spanning tree T not containing e. Then we prove that T is not a minimum spanning tree ok.

So, let this edge minimum cost edge connecting s and s complement. Let this s be u v, and this is the minimum cost edge connecting s s complement. So, suppose this is my s this part is s and this part is s complement. And this is u and this is v. So, this is e. So, where u belongs to s and v belongs to s complement. Now since T is a spanning tree, let me draw a spanning tree something some orbit this is a spanning tree for example, just to explain. So, this red tree is the spanning tree.

Suppose well, then since T is a spanning tree u and v must be part of this tree right. Because it contains all the vertices it contains it contains an unique path unique path from u to v which together with u v together with e, which is u v forms a cycle. See the red thing is the spanning tree. And since it is a spanning tree it includes all the vertices. So, it will includes u and v also.

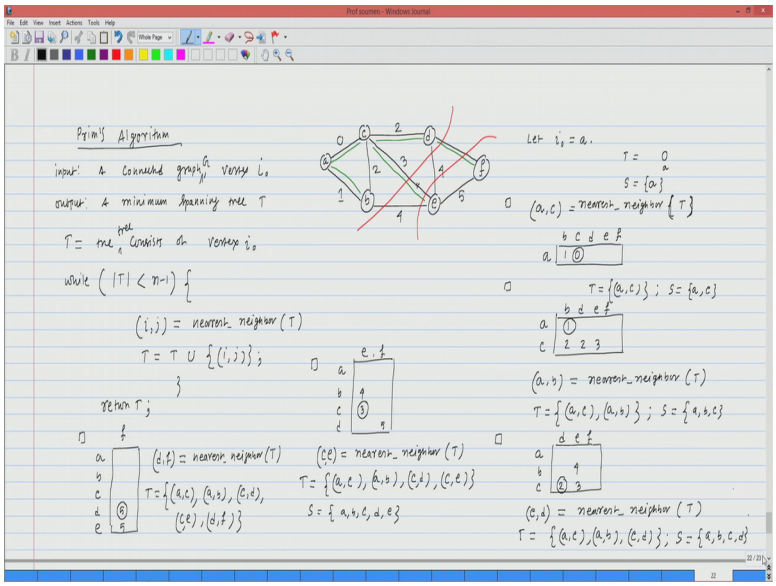
Now, in a tree between every pair of vertices for say u v there is a unique path. So, this is the unique path from u to v. And now e is not part of this tree. So, along with this e it forms a cycle. So, this is what this line says. So, this path so, I am talking about the path this one the path which is from u to v this path, this is the path unique path from u to v in the spanning in the spanning tree t. So, I am talking about this path means this green path this path has to include another edge f connecting s and s complement. This is true because at this at least for this example you can see that this part includes another s that s the edge f is this one. This is the edge I am talking about.

So, this is the edge, which is connecting s and s complement and this edge is there in the minimum spanning tree or in this a green path. So, this is what the f is. Now we will form another tree which has cost less than T, that tree is T plus now I will include e in the spanning tree and remove f, T plus e minus f is another spanning tree, and has less cost than T right. Because now what I am doing is that I started with this red tree and then I know that this is also a edge connecting s s complement.

But among all the edges which are connecting s s complement e is the minimum cost. Now what I do is that I remove this edge from this red tree and then I include this edge in the red spanning tree, instead of f remove f and add e and this tree has less cost, because the cost of e is less than or equal to the cost of f. Because e is the minimum cost edge connecting s and s complement. So, we got another spanning tree which is which has less cost than t. So, we prove that T is not minimum spanning tree. This is what we wanted to prove.

Now, the significance of this is that you can see that in the previous algorithm of every time of so this algorithm gives a way of defining s.

(Refer Slide Time: 09:15)

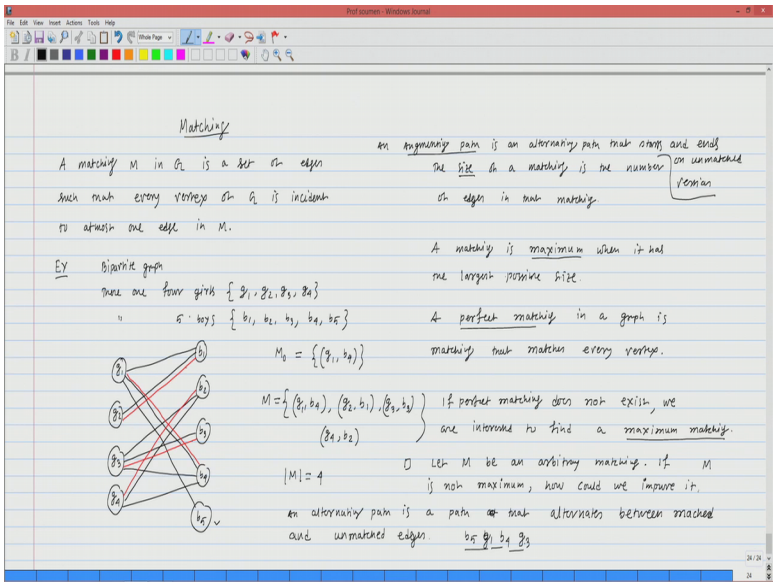


So, s is the tree edges at any stage and s complement of course, v minus s and what this nearest neighbor operation what it does is that it just identify the minimum cost edge between s and s complement of this cut. And that edge is included in the set of tree edges. So, this is how this Prim's algorithm sort of ensured that at every step whatever partition you have you the partition, you go you get from the previous or existing tree. And you at the next step you include an edge which is the minimum cost edge.

In the cut s s complement or minimum cost edge connecting s and s complement. So, this is what if all that is what I wanted to say about Prim's algorithm next we move to the matching in a graph. So, we learn what is matching and we spend significant amount of time in matching. So, first today we will talk about how to find maximum matching in bipartite graph using augmenting path algorithm.

So, let us start with a matching.

(Refer Slide Time: 11:11)



So informally speaking a matching in a graph is a collection of disjoint edges or independent edges sometime, we call let me give the formal definition a matching M in G the graph G is a set of edges, such that every vertex of G is incident to at most one edge in M. Let me explain because the definition should be very clear what is matching in a graph, let me start with the bipartite graph. And for the practical point of view I considered this example.

Suppose there are 4 girls. So, there are a 4 girls call them g 1 g 2 g 3 g 4. And there are 5 boys and call them b 1 b 2 b 3 b 4 b 5. Now I construct a bipartite graph, on the left part I have all the girls 4 girls. So, g 1 g 2 g 3 g 4 and on the right hand right part, because bipartite graph. So, partition the vertices into 2 parts, you have b 1 b 2 b 3 b 4 b 5. And there will be an edge between b I and b j if they are tend to each other or they like each other.

So, g 1 b 1; that means, girl g 1 likes a boy b 1. And similarly the girl g 1 also prefers b 4 and b 5. Girl g 2 prefers only b 1. Girl's g 3 sorry, girl g 3 prefers b 2 b 3 and b 4. And g 4 prefers b 2 and b 4. Now what is a matching? So, matching means it is a it is a matching just I mean you cannot match one girl with 2 boy. So, one girl will be matched with one boy similarly you cannot match 2 girls with one boy. So, that is why this is that it is a collection of edges such that every vertex of G is incident to at most one edge in M.

So, let me give one example of a matching here. One matching could be g 1 b 4 g 1 b 4. So, g 1 is matched with boy 4 and g 2 is matched with boy 1 and g 3 is matched with boy 3. And g 4 is matched with boy 4. So, here this is the matching. So, matching here the M the matching that I talked about is consists of 4 edges. That is see you cannot match g 1 with b 2 because g 1 does not prefer b 2. So, you have to choose find out a subset of this edges you can not just you know arbitrarily assign g 1 or match g 1 with b 2.

So, the matching M here is g 1 b 4, I am just omitting the comma here of g 2 b 1 g 3 b 3 and g 4 b 2. So, this is this is a matching. And it satisfy the property of the matching that every vertex now you can see that every vertex is incident to at most one edge; so this one edge in M. So, these are the edges in M now you can see that b is incident to only this edge b 2 is incident to only this edge, I am talking about the red edges only b 3 is incident to this edge b 4 is incident to this edge b 5 is not incident to any matched a So b So, here b 5 is at actually unmatched it left unmatched.

All the girls are matched of course, there are 4 girls you can match with 4 boys only. So, one boy extra is. So, so he the b 4 we call b 4 is not matched. And the red edges are called the match matching edges and the other edges are called a non-matching edges. So, these are the matching edges. So, the size of this matching the size of a matching is the number of edges in the matching. So, the size of this matching is equal to 4. So, formally the size of a matching is the number of edges in that matching. And a matching is maximum or a matching is a maximum matching when it has the largest possible size, yes.

So, here you it is not difficult to realize that this is a maximum matching because you can not include another edge. So, all the girls are matched. So, you can not include another edge. Of course, this is also a matching if I say M naught M naught is just for example, just g 1 g 1 b 4 this is also a matching, but this is not a maximum matching whereas, this is a maximum matching; so the optimization problem here to find maximum matching, because finding minimum matching is a trivial problem and a perfect matching in a graph is a matching that matches every vertex right.

So, this is not a perfect matching because not every vertex is matched here, b 5 is not matched. So, this is not a perfect matching, but this is a maximum matching also it is not difficult to realize that in order to get perfect matching the number of vertices in the

graph must be even. So, here the number of vertices is 4 plus 5 9. So, it is a odd number. So, you will never get a perfect matching it is very obvious. So, we have a maximum matching here, but not perfect matching ok.

So, if a perfect matching does not exist we are interested to find a maximum matching; obviously So, here the optimization problem is to find given a bipartite graph or any general graph you slowly we will learn the problem is to find the maximum matching. Next we learn a few definitions. One is called alternating path and augmenting path. So, let M be an arbitrary matching.

So, we can think of say we start with any matching, and the question here is that if M is not maximum how could we improve it? You start from arbitrary matching the ultimate goal is to find maximum matching you start from an arbitrary matching and then you improve it to get a maximum matching. Next we talk about of one 2 definitions one is called alternating path, and alternating path is a path that alternate between matched and unmatched edges unmatched edges ok.

So, I can probably give an example here. So, this is an alternating path. So, b b 5 b 5 to g 1 sorry, b 5 to g 1 this is a unmatched edge. And then from g 1 to b 4 b 4 this is a matched edge and then b 1 sorry b 4 to g 1 b 4 to g sorry g 3. So, this is unmatched this is matched this is unmatched. So, this is a alternating path from b 5 to g 3 yep. And augmenting path an augmenting path, sorry about this an augmenting path is an alternating path that starts and ends on unmatched vertices ok.

So, this is what the definition of a alternating path which is very important. I can not give an example of alternating path at this moment because this there is no alternating path probably with respect to this matching. So, this path is an alternating path, but this is not an augmenting path, because the alternating augmenting path should start with the unmatched vertex, and also end at the unmatched vertex, but g 3 is not an unmatched vertex is a matched vertex.

So, this is an alternative path, but this is not an augmenting path. Maybe in the next lecture we talk about augmenting path in details. And we keep an augmenting path algorithm to find a maximum matching in a bipartite graph. So, the augmenting path algorithm starts with an arbitrary matching or even in all matching there is no edge in the matching at the beginning. And then at every step it will find an augmenting path with

respect to the current matching, and improve the matching size by one at each step. And the algorithm will stops we will stop when there is no augmenting path with respect to a given matching.

So, augmenting path alternating path everything is with respect to a given matching, because it is the edges alternate between matching and unmatching edges. So, we learn about augmenting path algorithm to find maximum matching in bipartite graph in the next lecture.

Thank you very much.